Semantic Execution Environment - SEE

Michal Zaremba
Semantic Technology Institute (STI)
Innsbruck, Austria

Overview

- Semantic Execution Environment (SEE)
  - Vision
  - Mission
  - Goals
- Reference Implementation of SEE – Web Services Execution Environment (WSMX)
  - Architecture
  - Execution Semantic Implementation
  - Sample Components

OASIS SEE Overview

- SEE is originally based on idea of WSMO conceptual model
- OASIS SEE TC has evolved from the WSMX WG
- OASIS SEE TC – conceptual architecture
- WSMX WG – reference implementation
- SEE is aligned with OASIS SOA-RM
  - Current focus: Reference Ontology for Semantic Service Oriented Architectures

OASIS SEE TC

Started: November 2005

- After successful tutorial on SWS at OASIS Symposium in San Francisco, USA

Chairs:

- Michal Zaremba (STI Innsbruck)
- John Domingue (OU)

Members:

- LFUI, NUIG, OU, SAP AG, National Information Society Agency, CEFRIEL, CA Labs and several others active and passive contributors and members

Semantic Execution Environment – SEE Vision

The technology of Semantic Web Services (SWS) envisions:

- Easy access to various systems
- Seamless integration of heterogeneous entities
- Ad-hoc cooperation between various business parties
- Dynamic collaborations on the Web

SEE Vision:

Provide guidelines, justifications and implementation directions for an execution environment for Semantic Web Services

Semantic Execution Environment – Mission and Goals

SEE Mission:

Define an execution environment capable of managing all the aspects related to semantically enhanced Web services, to enable their discovery, selection, mediation and execution

Goals:

- Provide a WSMO testbed
- Demonstrate the viability of using WSMO as a means to achieve dynamic inter-operation of Web services

Life cycle

Publishing Create & publish Web service description

Discovery Determine usable services for a request

Composition Combine services to achieve a goal

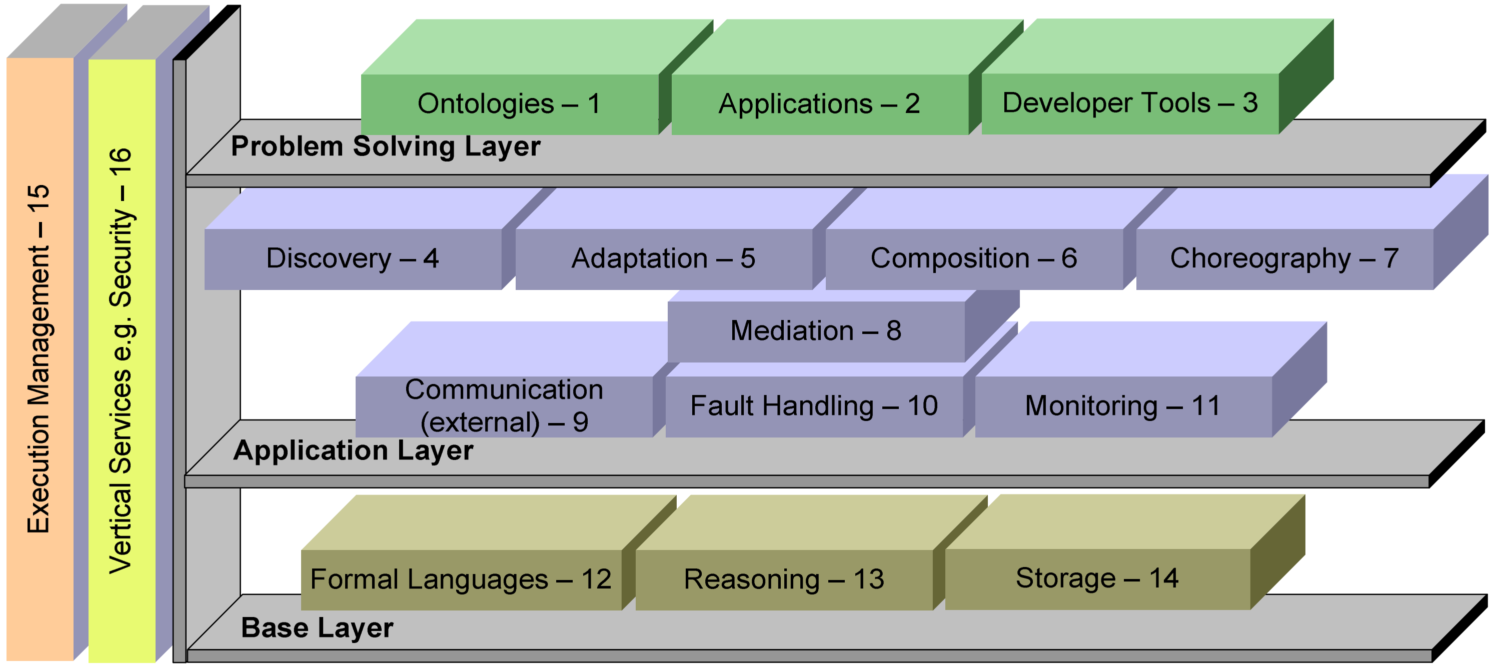
Selection Choose most appropriate service among the available ones

Mediation Solve mismatches (data, protocol, process) that hamper interoperation

Execution Invoke Web services following programmatic conventions

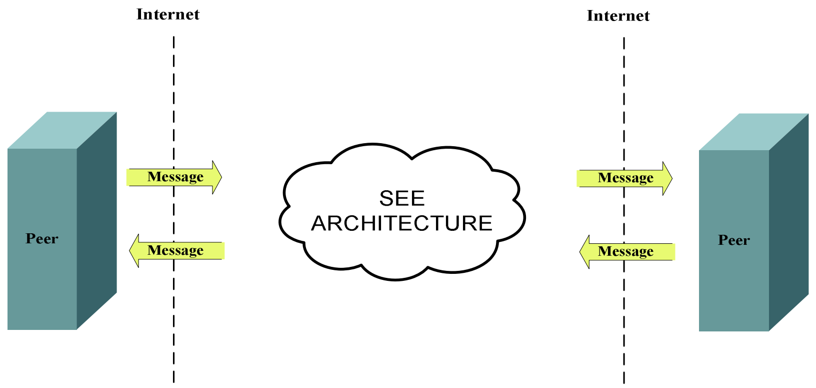
Semantic Web Services Infrastructure

- Service Oriented Architecture.
- Reference implementation for WSMO (WSMO is a conceptual model)



Usage scenario

- A P2P network of SEE ‘nodes’ and components
- Each SEE node and component described as a SWS
- Communication via WSML
- Distributed discovery – first aim
- Longer term aim - distributed execution environment



System entry points

storeEntity(WSMOEntity):Confirmation

- provides an administration interface for storing any WSMO-related entities (Web Services, Goals, Ontologies)

achieveGoal(Goal, OntologyInstance):Confirmation

- service requester expects WSMX to discover and invoke Web Service without exchanging additional messages

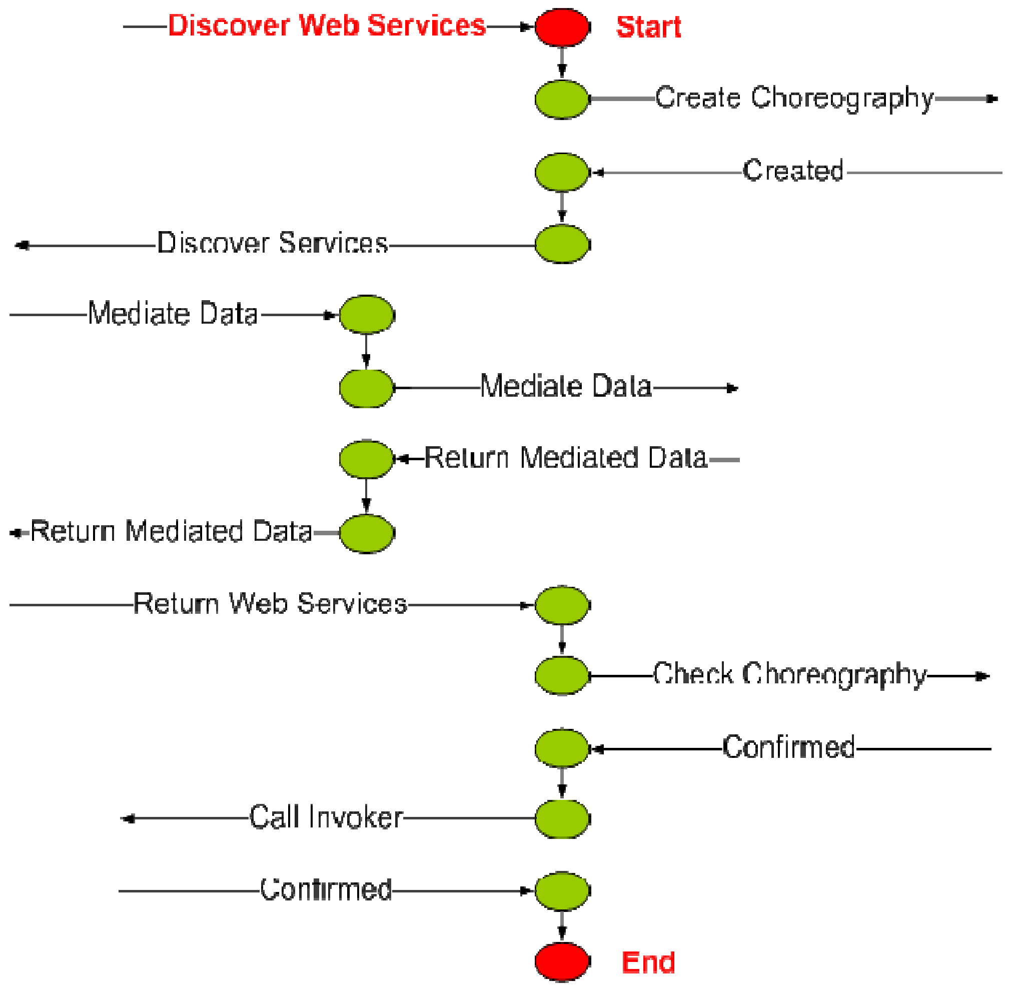
receiveGoal(Goal, OntologyInstance, Preferences): WebService[]

- list of Web Services is created for given Goal requester can specify the number of Web Services to be returned

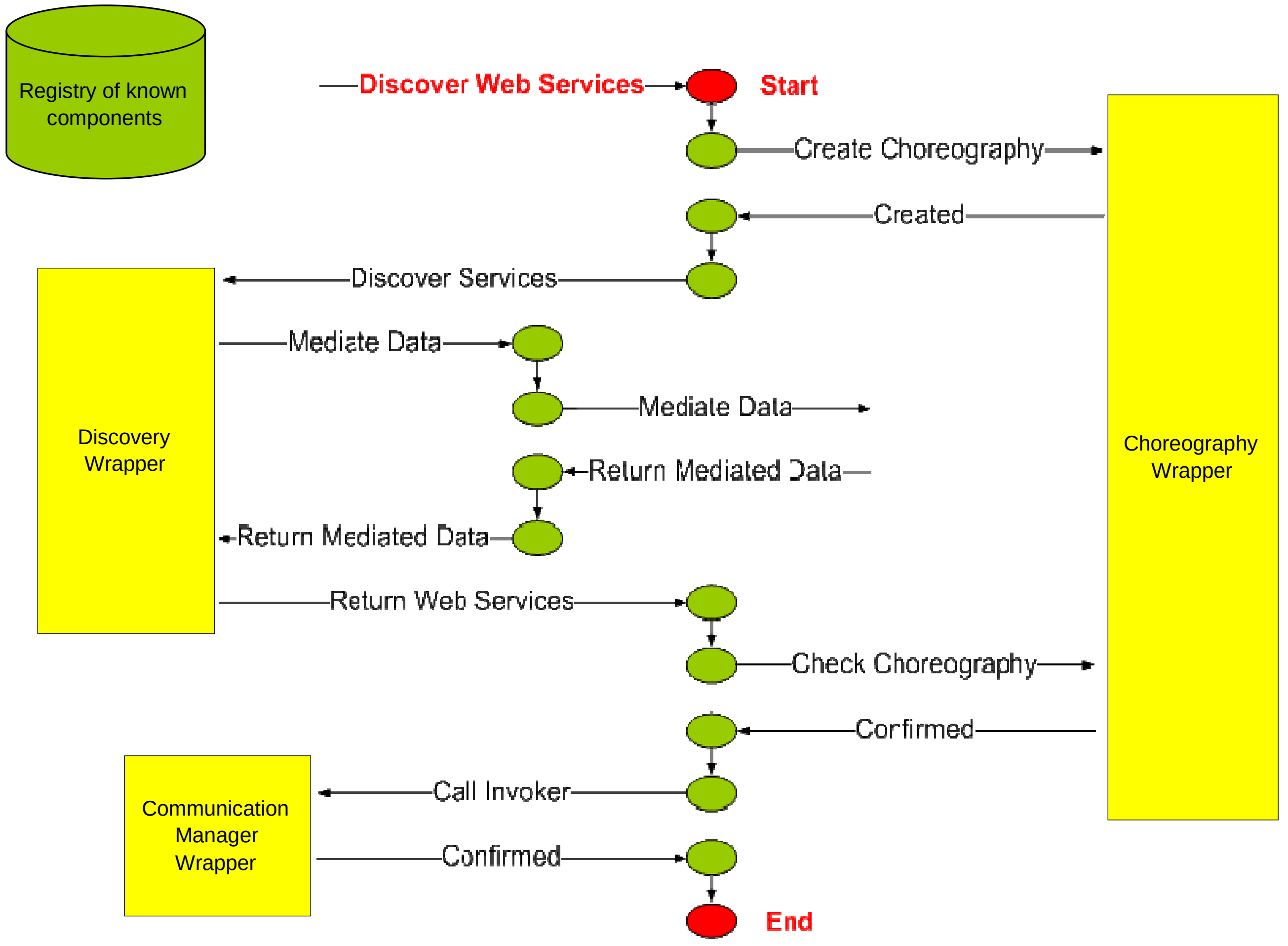
receiveMessage(OntologyInstance,WebServiceID, ChoreographyID):ChoreographyID

- back-and-forth conversation to provide all necessary data for invocation involves execution of choreographies and process mediation between service interfaces

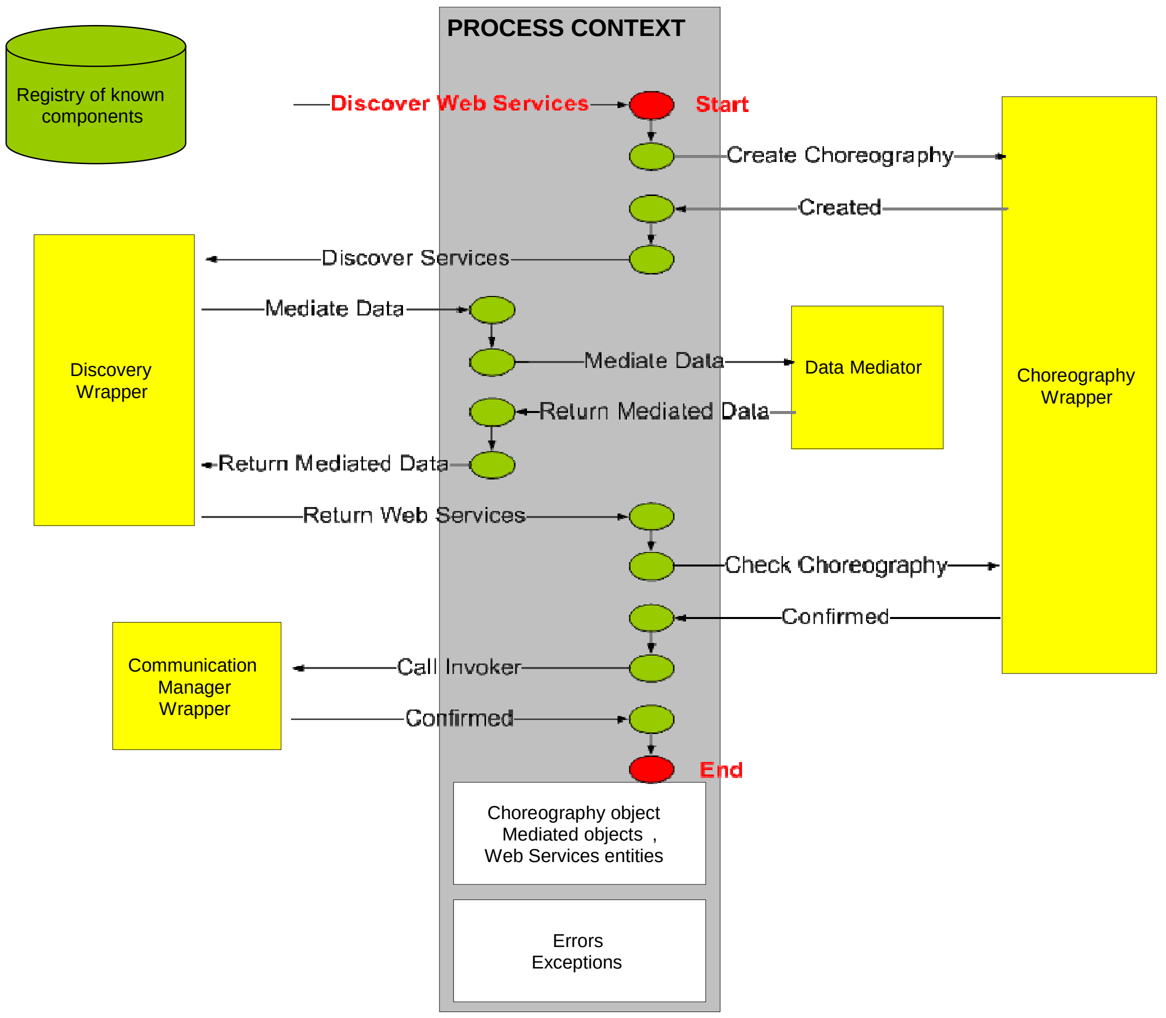
Execution semantics
Define “Business” Process (abstract)



Execution semantics
Generate Wrappers for Components (instantiation)

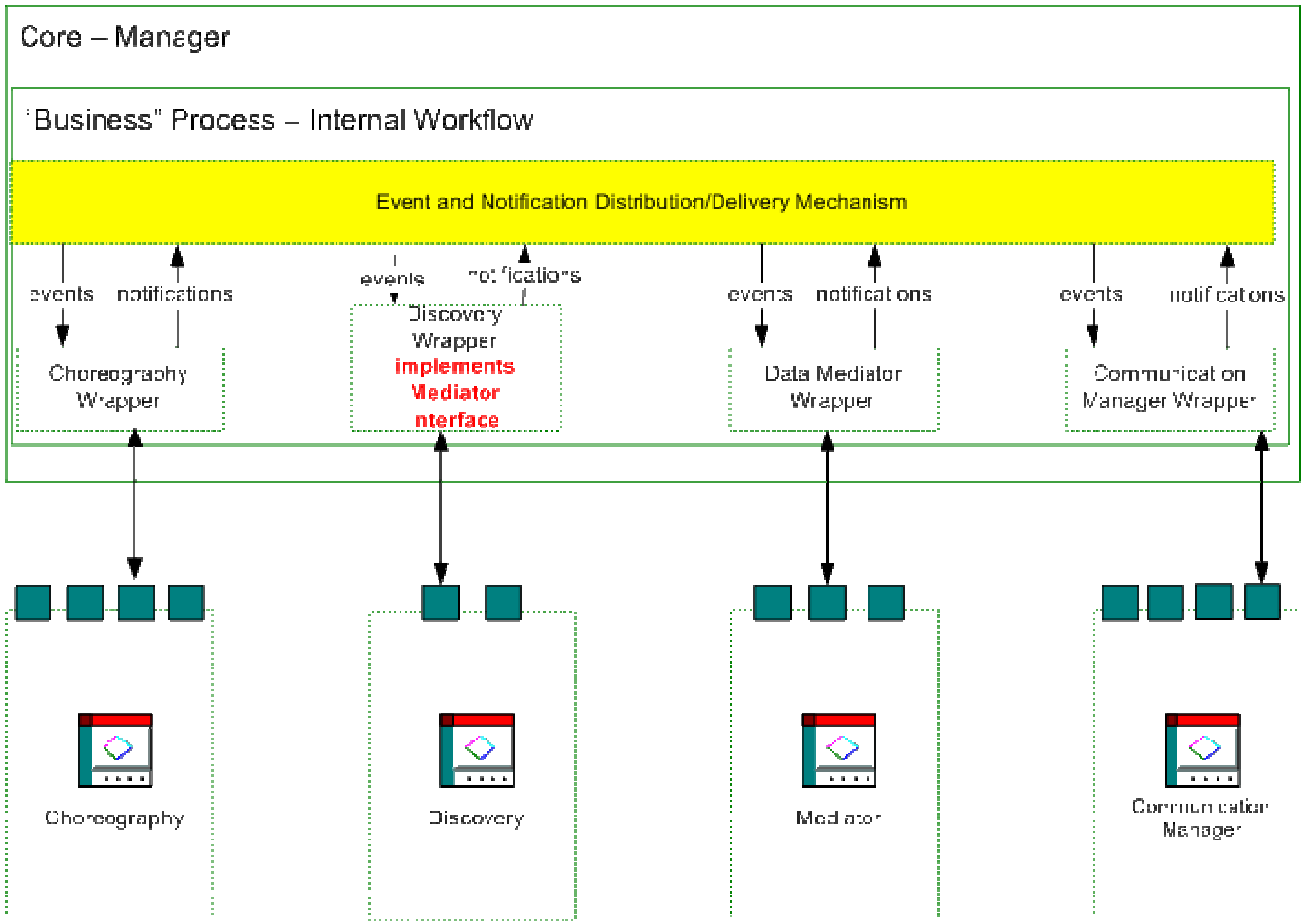


Execution semantics Context Data (in WSMX)



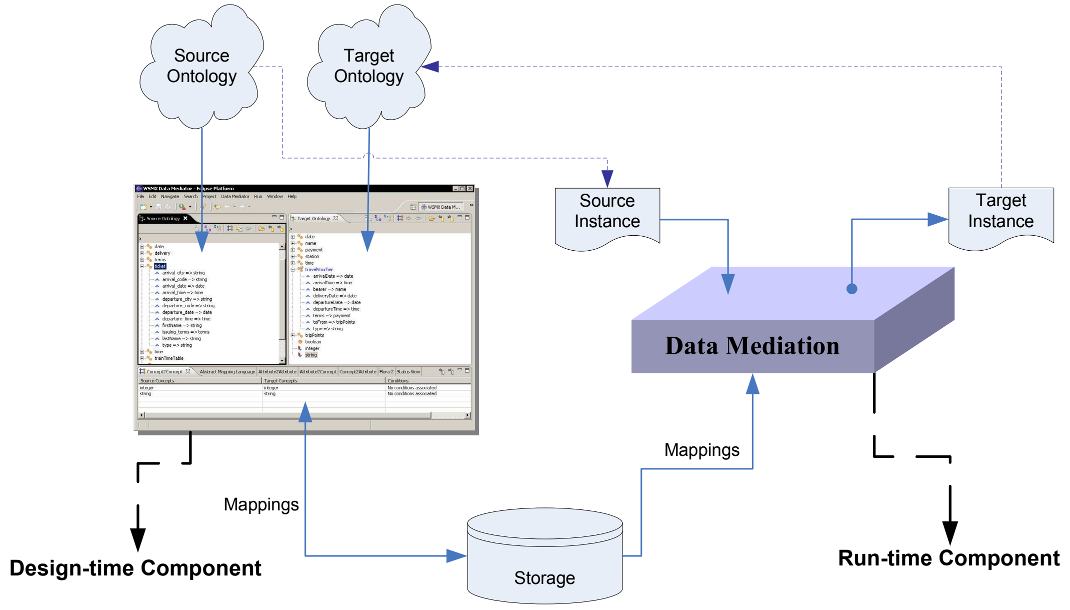
SOA4All contributing to NESSI

Event-based implementation (in WSMX)



Selected Component - Data Mediation

- Ontology-to-ontology mediation
- A set of mapping rules are defined and then executed
- Ontology Mapping Language
- Initially rules are defined semi-automatic
- Create for each source instance the target instance(s)



Selected Component - Data Mediation Design-time and Run-time mediation

Design-time

- Inputs
  - Source Ontology and Target Ontology
- Features
  - Graphical interface
  - Set of mechanism towards semi-automatic creation of mappings
  - Capturing the semantic relationships identified in the process
  - Storing these mappings in a persistent storage
- Output
  - Abstract representation of the mappings

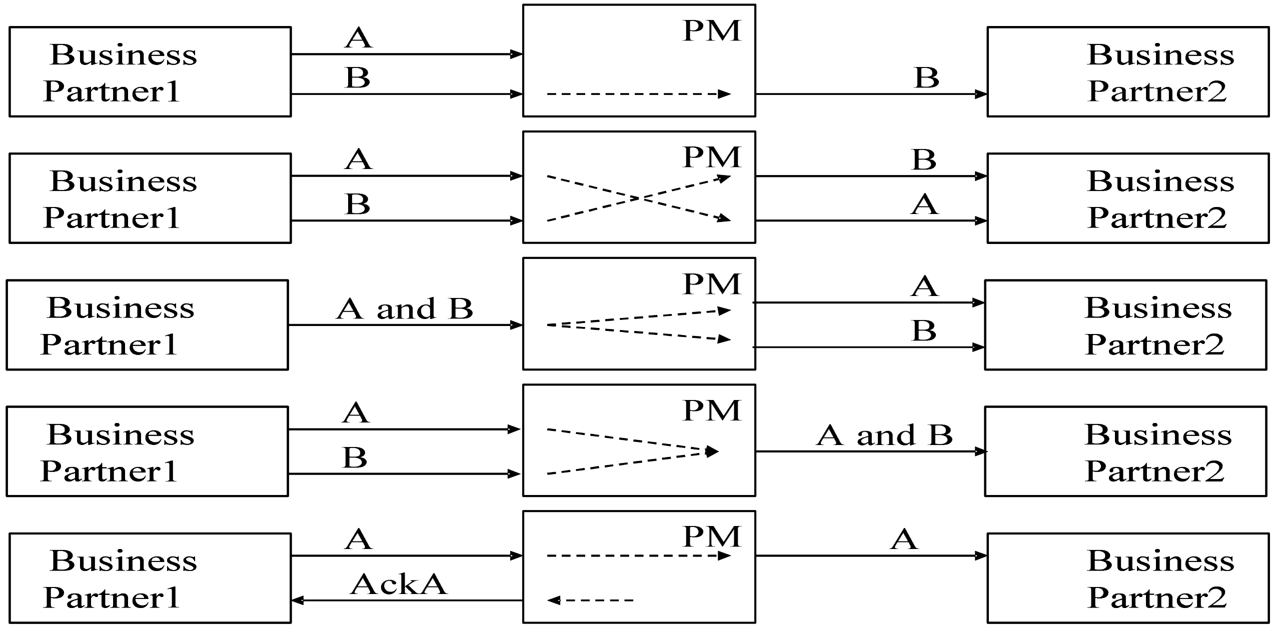
Run-time

- Main Mediation Scenario: Instance Transformation
- Inputs
  - Incoming data
    - Source ontology instances
- Features
  - Completely automatic process
  - Grounding of the abstract mappings to a concrete language
    - WSML
  - Uses reasoner to evaluate the mapping rules
- Outputs
  - Mediated data
    - Target ontology instances

Selected Component - Process Mediation

- Requester and provider have their own communication patterns
- Only if the two match precisely, a direct communication may take place
- At design time equivalences between the choreographies' conceptual descriptions is determined and stored as set of rules
- The Process Mediator provides the means for runtime analyses of two choreography instances and uses mediators to compensate possible mismatches

Selected Component - Process Mediation Addressed Mismatches



Conclusions

- Semantic Execution Environment (SEE) and Web Services Execution Environment (WSMX) are having the same conceptual model, which WSMO
- End to end functionality for executing SWS
- Formal execution semantics
- Real implementation available
- SEE Technical Committee hosted by OASIS, WSMX development effort hosted at SourceForge